Remarks/Arguments:

The above amendment modifies the specification in direct response to requests for correction in the 3/30/2007 Office Action (OA). The amendment corrects the format and language of the abstract, as requested in the OA.

Additionally, the amendment replaces the claims to define the invention more specifically to overcome the objections under 35 U.S.C. 103(a) of the OA, and to be patentable over prior art. Specifically of the previous Claims 1-20:

- Claims 2, 8, and 20 were restricted and withdrawn by the OA, and Applicant may choose to make a divisional patent application for these claims.
- Claim 3 has been amended to include more specificity from former claim 4.
- Claim 4 has been canceled.
- Claim 5 has been amended to be dependent on Claim 3, instead of Claim 4.

Summary of References

The OA includes the following two references:

Omoigui, US Patent Publication No. 2003/0126136, discloses a system and method for knowledge retrieval, management, delivery and presentation. This reference was discussed in the previous OA and in Applicant's previous Amendment C.

Popa, US Patent No. 6,567,802, dated May 20th, 2003, discloses a systematic approach to query optimization.

Popa does not Predate the Invention

The Application (filed December 22, 2003), and Amendment A to the Application, both include the cross-reference:

> This application is entitled to the benefit of Provisional Patent Application #60/436,441 filing date 12/26/2002.

The current Application contains no new matter over the above provisional patent application, as certified by the Applicant, and is entitled to the invention date of 12/26/2002. Therefore, the Popa patent dated 5/20/2003 does not predate the invention, and is not relevant prior art. The applicant also submits that the current Claims are non-obvious and therefore should be allowed, as discussed below.

Specific and Novel Planning Methods Distinguish Claims 3 and 5

Claims 3 and 5, as amended, describe two novel and beneficial elements of a particular mechanism to plan the execution of a query. This mechanism uses two phases to produce a procedural (step-by-step) execution plan from a declarative specification (unordered list of constraints):

1. In the first phase, a subset of the clauses in the query specification are selected to "generate" query variables.

2. In the second phase, all the clauses are ordered in a way that will optimize execution.

The following remarks will detail how each of these phases are novel and unobvious, and overcome objections to the original Claims 4 and 5.

Objection to Claim 4 Overcome

The cited sections of Omoigui ([1117], [1125]), disclose allowing the user to specify certain numerical conditions on acceptable result documents, for example the number of articles in an email "thread", or minimum and maximum acceptable ages for a document. Omoigui does not disclose using these or any other numeric attributes for ordering the steps of a search or computation. Omoigui also does not disclose a comparable two-phase algorithm for planning, or any other particular algorithm for planning, other than to translate queries into the well-known "SQL" language for execution.

Claim 4 is the first step of the two-step planning phase. This first step picks which clauses will be used to generate ("produce") variable values. All other clauses which mention this value will simply use ("consume") it. The specific numeric attribute used for selection in Claim 4 is the number of distinct variable value results ("branching"). Further, this branching is re-evaluated after each selection of a clause in the planning process. This produces the unexpected benefit of dramatically increasing the efficiency of the final plan (because it can iterate over a smaller number of possible values), and of providing additional opportunity for optimization in a second ordering phase (because the clauses can still be re-ordered) without inordinate computation during the planning phase (because we do not have to consider every possible ordering of the clauses). Thus the "pick next generating clause based on minimum branching" rule of Claim 4 is both novel and unobvious.

Objection to Claim 5 Overcome

The further cited sections of Omoigui ([0072], [0322], [0341]) mention the ability to optimize

user input, presentation, and retrieval. None of the specific mechanisms for this optimization are described, save for the use of "natural language" processing for information retrieval. None of these disclose the particular numeric attribute and ordering process described in Claims 4 and 5. Using a numeric attribute based on both cost and branching provides the unexpected benefit of efficient computation (because clauses that are more expensive to compute are executed less often, because they are executed before branching can cause additional backtracking and re-evaluation), while remaining also efficient to plan.

Because each phase of the described two-phase planning method is novel and unobvious, Claims 4 and 5 as amended, and the dependent claims 6, 15, 16 and 19 should all be allowed.

Dynamic Dispatch Applied to Semi-Structured Data Distinguishes Claim 6 and Claim 19

Omoigui ([0865], [0899]) discloses methods of following semantic links, where the SQML eventually returned may depend on the dynamic value of a variable object. Claim 6 and Claim 19, as amended, specify a particular method for "object-oriented" dynamic dispatch, in which the labels of semi-structured data values (e.g. the element tag name and namespace of an XML element) are used to determine a location to look for a clause definition. Omoigui does not disclose any dispatch based on the type of an object, much less the specific use of the labels of semi-structured data for this purpose. (Omoigui dispatches based on object storage location).

While object-oriented dynamic dispatch is common in computer programming languages, Claim 6 and Claim 19 represent a novel technique for applying object-oriented dispatch to semi-structured data. This technique unexpectedly provides the benefits of object-oriented dispatch

(e.g. polymorphism and inheritance) without requiring the addition of any type information beyond that already present in semi-structured data.

Thus Claim 6 and Claim 19, as amended, are novel, the technique unobvious, and should be allowed.

Method of Making Data Changes Compatible with Generate-and-Test and User-Specified Clauses Distinguishes Claim 15

Claim 15, as amended, both define a specific method for combining declarative specifications with clauses, specifying data changes with some of the clauses, and implementing generate-and-test for evaluation of those specifications. This particular combination is novel and unobvious, as discussed below.

Omoigui discloses a number of ways in which data storage can be changed, including via email messages, via SQL-based databases to store cached source data, and via the storage and updating of discovered semantic data. Omoigui also discloses a declarative language, SQML, which specifies desired results and an evaluation of those results by translating to a database query language like SQL. However, Omoigui does not disclose any method, much less the method of Claim 15, to specify arbitrary data changes within SQML. Thus Claim 15 is not disclosed by Omoigui.

In traditional data management systems, changes to data are described by the user in either

1. a purely procedural manner (e.g. step 1: set x=3; step 2: store x at location a; step 3: read y from location b...), or

2. a declarative if-then rule (e.g. IF x=3 and location a contains 4 THEN store x at location b.).

The first example is in the style of procedural languages like "C", "Perl", and the second example is in the style of an SQL database (which uses UPDATE to indicate the "THEN" portion, and WHERE to indicate the "IF" portion of the rule). The data changes caused by execution of these descriptions can be called "side-effects" of the execution.

A procedural language allows for user-specified clauses and for data changes, but does not provide generate-and-test, since statements can not be "undone" and "redone". This lack of generate-and-test means that the system cannot automatically search for desired results, rather the user must specify the step-by-step algorithm to produce the results. Thus Claim 15 are not anticipated by any procedural languages, including those mentioned by Omoigui.

A traditional declarative language, e.g. for an SQL database, allows for generate-and-test search through the database, which occurs as the database searches through the data for combinations that satisfy the "IF" or "WHERE" clause. An SQL database may provide for user-defined clauses, however these clauses must be either wholly free of any data changes (e.g. database "views"), or are procedural units that can only be included in the "THEN" portion of a definition (e.g. "stored procedures"). A traditional declarative language does not allow for user-defined clauses to include data side-effects and be used within the powerful generate-and-test engine. Thus traditional declarative languages, including the ones mentioned by Omoigui, do not anticipate Claim 15.

That the mechanism of Claim 15 is unobvious is demonstrated by:

a) it is contrary to conventional design of either procedural or declarative systems,

b) it has not been implemented in Omoigui or anywhere else to the knowledge of the applicant,

c) it provides the benefit of using a single language to describe both arbitrary data changes and arbitrary data searches, and

d) it allows the intermixing of side-effects and data queries into reusable component clauses.

Thus Claim 15 as amended is both novel and unobvious, and Claim 15 and the dependent Claims 16, and 19 should be allowed.

Method of Updating Indexes Distinguishes Claim 16

Omoigui discloses caching of intermediate query results and indexing information by meta data and semantics. Also SQL databases are well-known to index columns of data tables. The caching and indexing method claimed in Claim 16 is different, because it specifies a novel means for updating the indexes when underlying data changes.

Omoigui does not directly describe any mechanism for updating indexes, much less the specific mechanism of Claim 16. Traditionally, query systems will simply put a time and date on cached results, and recalculate the results after a set period of time. One disadvantage to this method is that if the data changes during the set period, the cache becomes "stale", and invalid results may be returned. The other disadvantage is that if the data has not changed, then an unnecessary recalculation is done after the set period expires.

SQL and other databases, mentioned in Omoigui, also have various caching and indexing

mechanisms, typified by either a *column index*, a *query plan cache*, or a *query result cache.* We will discuss each in turn in order to contrast with the invention:

- A column index is efficiently partially recalculated when specific rows of a data table are changed. The limitation of a column index is that it can only cache values of a particular column, not of an arbitrary query.
- A query plan cache stores execution plans for queries, so that subsequent uses do not have to plan again. The limitation of these caches is that the query plan must be re-executed each time the query result is needed.
- A query result cache may store the full result of a query. The limitation of a query result cache is that either it is not updated when data changes, causing potentially stale results, or an entire cache associated with a particular table must be (inefficiently) recalculated when even the smallest portion of the data in the table is changed.

The mechanism of Claim 16 is novel, because when data changes, it uses a re-execution of a modified version of a query plan to discover which portions of the cache and index need to be modified. This novelty is also unobvious, as demonstrated by the unexpected benefits that it:

a) eliminates the risk of stale results being returned,

b) allows arbitrary queries to be cached for quick execution, and

c) minimizes the amount of updates required to the cache and index.

Thus Claim 16 as amended and the dependent claim 19 are novel and unobvious, and should be

allowed.

Conclusion

For all the above reasons, applicant submits that the specification and claims are now in proper form, and that the active claims all define patentably over the prior art. Therefore he submits that this application is now in condition for allowance, which action he respectfully solicits.

Conditional Request for Constructive Assistance

Applicant has amended the specification and claims of this application so that they are proper, definite, and define novel structure which is also not obvious. If for any reason this application is not believed to be in full condition for allowance, applicant respectfully requests the constructive assistance and suggestions of the Examiner pursuant to M.P.E.P. Section 2173.02 and Section 707.07(j) in order that the undersigned can place this application in allowable condition as soon as possible and without the need for further proceedings.

Very respectfully,

Judson Ames Cornish

646 Georgia Avenue

Palo Alto, CA 94306

650-533-0835 (Mobile)

650-331-1402 (Daytime)

650-494-2639 (Evenings)

815-301-9795 (Fax)

ames@cornishes.net